```
class Graph:
    def __init__(self, vertex):
        self.V = vertex
        self.graph = []

    def add_edge(self, u, v, w):
        self.graph.append([u, v, w])

    def find(self, parent, i):
        if parent[i] == i:
            return i
        return self.find(parent, parent[i])

    def union(self, parent, rank, x, y):
        xroot = self.find(parent, x)
        yroot = self.find(parent, y)

        if rank[xroot] < rank[yroot]:
            parent[xroot] = yroot
        elif rank[xroot] > rank[yroot]:
            parent[yroot] = xroot
        else:
            parent[yroot] = xroot
            rank[xroot] += 1

    def kruskal(self):
        result = []
        i, e = 0, 0

        self.graph = sorted(self.graph, key=lambda item: item[2])
        parent = []
        rank = []

        for node in range(self.V):
            parent.append(node)
            rank.append(0)

        while e < self.V - 1:
            u, v, w = self.graph[i]
            i = i + 1
            x = self.find(parent, u)
            y = self.find(parent, v)

            if x != y:
                e = e + 1
                result.append([u, v, w])
                self.union(parent, rank, x, y)

        for u, v, weight in result:
            print("Edge:", u, v, end=" ")
            print("-", weight)


g = Graph(5)
g.add_edge(0, 1, 8)
g.add_edge(0, 2, 5)
g.add_edge(1, 2, 9)
g.add_edge(1, 3, 11)
g.add_edge(2, 3, 15)
g.add_edge(2, 4, 10)
g.add_edge(3, 4, 7)
g.kruskal()}
```